

Demo - use case 1: Querying tables across levels, visualize data and store query

Moritz Mainka

2025-04-08

Introduction

The goal of this use case is to show how to query data across different levels (plot, sample, species) and visualize the data. We will use the `ggplot2` package for visualization and the `dplyr` package for data manipulation. This is just one way of handling data, the R language provides a myriad of other ways to do the same operations.

Load plot-info table

The first step is to load the plot-level information table. This table is central as it provides general information of plots. As a good practice, we start by assessing the table.

```
# load plot-level information table
plot_info <- read.csv("../data/plot_level_info.csv")

# as a first step, lets have a look at the table:
head(plot_info)
```

	plot_id	lat	long	site	environment	core	geology_code
1	AD_T_BP01	78.17369	15.88830	AD	T	core	G24
2	AD_T_BP02	78.17376	15.88909	AD	T	core	G24
3	AD_T_BP03	78.17356	15.88917	AD	T	core	G24
4	AD_T_BP04	78.17364	15.89029	AD	T	core	G24
5	AD_T_BP05	78.17198	15.89659	AD	T	gradient	G24
6	AD_T_BP06	78.17214	15.89606	AD	T	gradient	G24

	geology	plot_type
1	Shale+sandstone	BP
2	Shale+sandstone	BP
3	Shale+sandstone	BP
4	Shale+sandstone	BP
5	Shale+sandstone	BP
6	Shale+sandstone	BP

Next to `plot_id`, this table provides information on coordinates of the plots (lat-long), geology, environment (tundra (T), bird cliff (B), settlement (S)), etc. In total 173 rows = unique plots have been sampled.

Merge with other table on plot level

Let's do our first merge combining plot-level information with plot-level data. For this we will use the biomass data per plant functional type. Both tables are on the same level and therefore connected through one-to-one relationships. This means that each row in one table corresponds to one row in the other table.

```
# load data with plant cover per plant functional type
veg <- read.csv("../data/veg_pft_cover.csv")

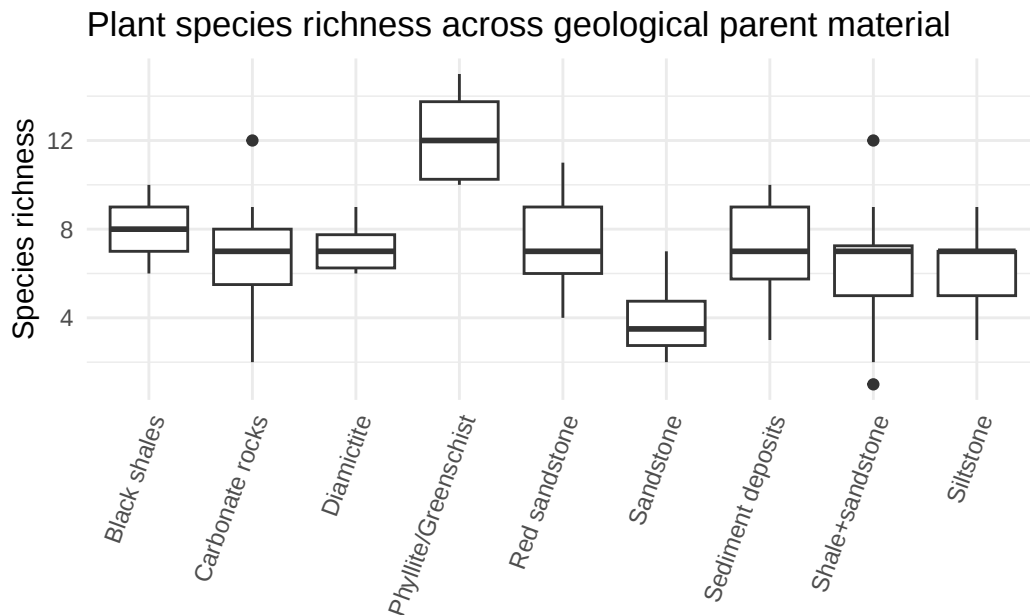
# merge plot-level information with plot-level data using the plot_id column
plot_data <- merge(plot_info, veg, by = "plot_id", all.y = T)
```

Our merge was successful. Out of all 173 existing plots, 142 have recorded vegetation cover data. In addition, species richness and Shannon diversity indices are provided.

Visualize plot-level data

```
library(ggplot2)

# plot species richness against the share of bare_ground
ggplot(plot_data, aes(x = as.factor(geology), y = species_richness)) +
  geom_boxplot() +
  labs(title = "Plant species richness across geological parent material",
       x = "",
       y = "Species richness") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 70, hjust = 1))
```



Combine plot with sample-level data

Now we want to combine the plot-level data with sample-level data. For this we will use the qPCR and soil chemistry data on sample-level.

```
# let's first remove what we don't need anymore from the environment
rm(veg, plot_data)

library(dplyr)

# load qpcr data
qpcr <- read.csv("../data/microbio_qpcr.csv")

# merge plot-level data with site-level data using the site_id column
plot_qpcr <- merge(plot_info, qpcr, by = "plot_id", all.y = T)

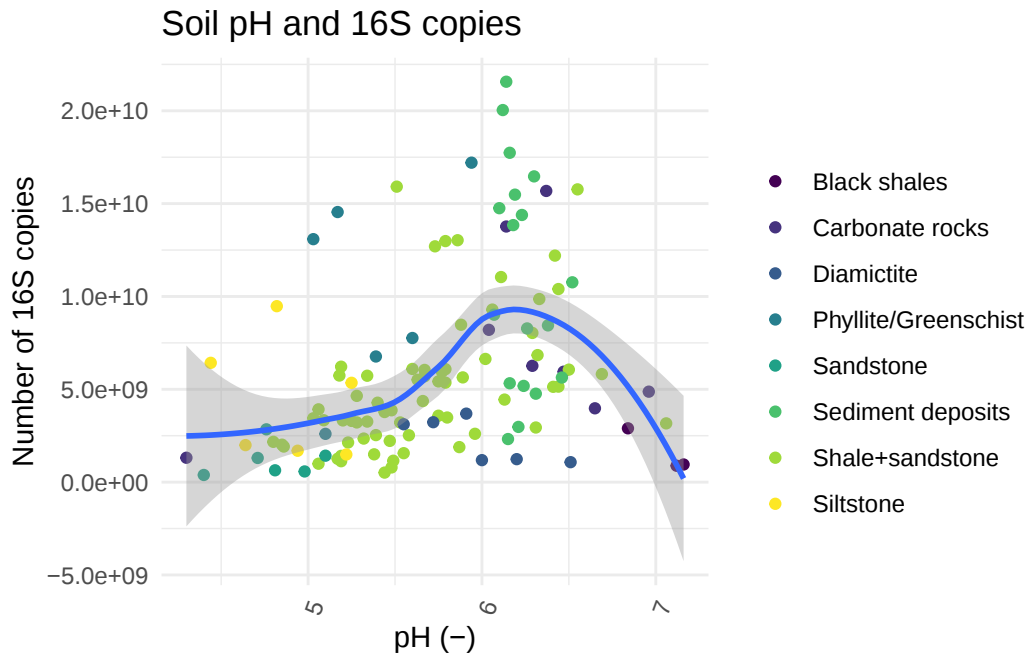
# let's now add the sample-level data

# load sample level soil chemistry data and select relevant columns
soil <- read.csv("../data/soil_chem.csv") %>%
  select(plot_id, sample_id, pH_CaCl2, CECeff_BaCl2)

# merge plot_data with soil sample data
query <- merge(plot_qpcr, soil, by = c("sample_id", "plot_id"), all.x = T)
```

Plot data

Now, let's plot the queried data. Let's generate a plot showing the relationship of qPCR results and soil pH.



It can be seen that there is an pH optimum for the number of 16S copies.

Save table as csv

To finally export a query of the data, we can use the `write.csv` function.

```
# save table as csv
write.csv(query, file = "../demo/query_demo1.csv", row.names = F)
```